Lazy Training of Radial Basis Neural Networks

José M. Valls, Inés M. Galván, and Pedro Isasi

Universidad Carlos III de Madrid - Departamento de Informática,
Avenida de la Universidad, 30 - 28911 Leganés (Madrid), Spain
jvalls@inf.uc3m.es

Abstract. Usually, training data are not evenly distributed in the input space. This makes non-local methods, like Neural Networks, not very accurate in those cases. On the other hand, local methods have the problem of how to know which are the best examples for each test pattern. In this work, we present a way of performing a trade off between local and non-local methods. On one hand a Radial Basis Neural Network is used like learning algorithm, on the other hand a selection of the training patterns is used for each query. Moreover, the RBNN initialization algorithm has been modified in a deterministic way to eliminate any initial condition influence. Finally, the new method has been validated in two time series domains, an artificial and a real world one.

Keywords: Lazy Learning, Local Learning, Radial Basis Neural Networks, Pattern Selection.

1 Introduction

When the training data are not evenly distributed in the input space, the non-local learning methods could be affected by decreasing their generalization capabilities. One way of resolving such problem is by using local learning methods[3,9]. Local methods use only partially the set of examples during the learning process. They select, from the whole examples set, those that consider more appropriate for the learning task. The selection is made for each new test pattern presented to the system, by means of some kind of similarity measurement to that pattern. k-NN [4] is a typical example of these systems, in which the selected learning patterns are the k closest to the test pattern by some distance metric, usually the Euclidean distance.

Those methods, usually known as lazy learning or instance-based learning algorithms [1], have the inconvenience of being computationally slow, and highly dependent on the number of examples selected and on the metric used, being frequent the situations where an Euclidean metric might not be appropriate.

Bottou and Vapnik [2] introduce a dual, local/non-local, approach to give good generalization results in non-homogeneous domains. This approach is based on the selection, for each test pattern, of the k closest examples from the training set. With these examples, a neural network is trained in order to predict the test pattern. This is a good combination between local and non-local learning. However, the neural network used is a linear classifier and the method assumes

S. Kollias et al. (Eds.): ICANN 2006, Part I, LNCS 4131, pp. 198–207, 2006.

that Euclidean distance is an appropriate metric. Besides, it considers that all test patterns have the same structure but some domains would require different behaviors when being in different regions.

In this work we introduce some modifications in the general procedure of [2], by considering the use of Radial Basis Neural Networks (RBNN)[6,5]. RBNN have some advantages when using dual techniques: they are non-linear, universal approximators [7] and therefore the metric becomes a non-critical factor; besides, their training is very fast, without increasing significatively the computational cost of standard lazy approaches.

We propose to use RBNN with a lazy learning approach, making the selection of training patterns based on a kernel function. This selection is not homogeneous, as happened in [2]; by opposite it is detected, for each testing pattern, how many training patterns would be needed, and what is the importance in the learning process of each one of them. This importance is taken into consideration, in the form of a weight, in the learning process of the network.

When a lazy approach is combined with RBNN, two important aspects must be taken into account. In one hand, the initialization of the RBNN training algorithm is a critical factor that influences their performance. This algorithm has been modified in a deterministic way to eliminate any initial condition influence. In other hand, it may occur that no training pattern is selected for certain test patterns, due to the distribution of data in the input space. In those case the system must provide some answer. We propose two different approaches to treat this problem.

The final method results to be a dual local non-local method, where the initialization of the network is deterministic and the method is able to determine the degree of locality of each region of the space, by means of a kernel function that could be considered as a parameter, and modified appropriately. In some cases a test pattern could be considered as non-local in the sense that it corresponds to more frequent situations. In this case almost the totality of the training patterns will be selected, and the method behaves like an non-local approach. This transaction between local and non-local behavior is made automatically.

2 Description of the Method

The learning method proposed in this work has been called LRBNN (Lazy RBNN method) and is based on the selection, from the whole training data, of an appropriate subset of training patterns in order to improve the answer of the network for a novel pattern. For each new pattern received or query, a new subset of training examples is selected. The main idea consists of selecting those patterns close to the new query instance, in terms of the Euclidean distance. In order to give more importance to the closest examples, a weighting measure that assigns a weight to each training example is applied. This is done by using a kernel function which depends on the Euclidean distance from the training pattern to the query. In this work, the inverse function ($K(d) = 1/d$, where d

is the distance from the training pattern to the new query) is used. A more detailed information about the use of this function can be found in [8].

To carry out this idea, a n-dimensional sphere centered at the test pattern is established, in order to select only those patterns placed into it. Its normalized radius (respect to the maximum distance from any example to the query), called r_r, will be used to select the training patterns situated into the sphere, being r_r a parameter that must be established before the application of the learning algorithm. Next, the sequential structure of LRBNN method is summarized.

Let $\mathbf{q} = (q_1, ..., q_n)$ be the query instance. Let X be the whole available training data set:$X = \{(\mathbf{x_k}, \mathbf{y_k})\ k = 1 \ldots N; \mathbf{x_k} = (x_{k1}, \ldots, x_{kn}); \mathbf{y_k} = (y_{ki}, \ldots, y_{km})\}$. For each $\mathbf{q}$,

1. The standard Euclidean distances d_k from the query to each training example are calculated. Then, the relative distance d_{rk} is calculated for each training pattern: $d_{rk} = d_k/d_{max}$, where d_{max} is the distance from the novel pattern to the furthest training pattern.
2. A kernel function $K()$ is used to calculate a weight for each training pattern from its distance to the query. This function is the inverse of the relative distance d_{rk}: $K(x_k) = 1/d_{rk}; k = 1 \ldots N$
3. These values $K(x_k)$ are normalized in such a way that the sum of them equals the number of training patterns in X. These normalized values are called normalized frequencies, f_{nk}.
4. Both d_{rk} and f_{nk} are used to decide whether the training pattern is selected and -in that case- how many times is included in the training subset. They are used to generate a natural number, n_k, following the next rule:

$$\text{if } d_{rk} < r_r \text{ then } n_k = int(f_{nk}) + 1 \text{ else } n_k = 0 \tag{1}$$

   At this point, each training pattern in X has an associated natural number, n_k, which indicates how many times the pattern (x_k, y_k) will be used to train the RBNN in order to predict the query q.
5. A new training subset associated to $\mathbf{q}$, X_q, is built up. Given a pattern (x_k, y_k) from the original training set X, it will be included in X_q if $n_k > 0$. Besides, (x_k, y_k) will be randomly placed n_k times in X_q.
6. The RBNN is trained using X_q: the neurons centers are calculated in an unsupervised way using K-means algorithm in order to cluster the input training patterns included in the subset X_q. The neurons widths are evaluated as the geometric mean of the distances from each neuron center to its two nearest centers, and the RBNN weights are estimated in a supervised way in order to minimize the mean square error measured in the training subset X_q.

In order to apply the learning method to train RBNN, two features must be taken into account: On one hand, the results would depend on the random initialization of the K-means algorithm which is used to determine the locations of the RBNN centers and must be applied for each query. On the other hand, when the test pattern is located in a region of the input space where the examples are scarce, it could happen that no training examples are selected. We present solutions to both problems, which are described next.

K-means initialization. Having the objective of achieving the best performance, a deterministic initialization, instead of the usual random ones, is proposed. The idea is to obtain a prediction of the network with a deterministic initialization of the centers whose accuracy is similar to the one obtained when several random initializations are done. The initial location of the centers will depend on the location of the closest training examples selected. The deterministic initialization is obtained as follows:

- Let $(\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_l})$ be the l selected training patterns, inversely ordered by their distance to the query instance. Let m be the number of hidden neurones of the RBNN to be trained.
- If $m \leq l$ then the center of the i_{th} neuron is initialized to the $\mathbf{x_i}$ position, for $i = 1, 2, \ldots, m$. Otherwise $(m > l)$, l neurones will be initialized to the $\mathbf{x}_i$ position, for $i = 1, 2, \ldots, l$, and the remaining $m - l$ neurones (lets call this number p) will be randomly initialized in the following way:
  - M_q, the centroid of the set X_q, is evaluated.
  - p centers $(c_{1q}, c_{2q}, ...c_{pq})$ are randomly generated, such as $\|c_{jq} - M_q\| < \epsilon$, $j = 1, 2, \ldots, p$, where ϵ is a very small real number.

Empty training set. It has been observed that when the input space data is highly dimensional, in certain regions of it the data density can be so small that the sphere centered at the query instance does not include any train pattern into it if the relative radius is small. When this situation occurs, an alternative way to select the training patterns must be taken. In our work, we propose two different approaches which are experimentally evaluated.

1. If the subset X_q associated to a query $\mathbf{q}$ is empty, then we apply the method of selection to the closest training pattern, as if it was the test pattern. Thus, the selected set will have, at least, one element.
2. If X_q is empty, then the network is trained with X, the set formed by all the training patterns. In other words, the network is trained as usual, with all the available patterns.

3 Experimental Validation

We have applied LRBNN to two domains, the Mackey-Glass and the Venice Lagoon time series. As it was remarked in section 2, the relative radius r_r must be given as an external parameter of the method in order to study its influence on the performance of the model. Besides, RBNN with different architectures - i.e. different number of hidden neurons- must be trained so that the influence of the network architecture can also be studied.

The method incorporates solutions regarding to the initialization of centers and the possibility of having empty training sets. These solutions are validated in the experiments where we have applied the lazy approach with both ways of initializing the centers: the random and the deterministic one. Moreover, in the cases where some test patterns can not be predicted because the associated training subset is empty, the approaches mentioned in section 2 have been applied.

3.1 An Artificial Time Series Prediction Problem: The Mackey-Glass Time Series

The Mackey-Glass time series is a well known artificial time series widely used in the literature about RBNN, [10],[6]. The data used in this work has been generated following the studies mentioned above. The task for the RBNN is to predict the value of the time series at point $x[t+50]$ from the earlier points $(x[t], x[t-6], x[t-12], x[t-18])$. 1000 data points form the training set, corresponding to the sample time between 3500 and 4499. The test set is composed by the points corresponding to the time interval $[4500, 5000]$. Both sets have been normalized in the interval $[0, 1]$. The proposed LRBNN method has been applied to this artificial time series, where RBNN of different architectures have been trained during 500 learning cycles varying the relative radius from 0.04 to 0.24.

Table 1. Mean errors with random initialization of centers. Mackey-Glass time series

	Hidden Neurones							
r_r	7	11	15	19	23	27	NP	%PP
0.04	0.02527	0.02641	0.02683	0.02743	0.02691	0.02722	45	91
0.08	0.02005	0.01891	0.01705	0.01571	0.01716	0.01585	0	100
0.12	0.02379	0.01954	0.01792	0.01935	0.01896	0.01940	0	100
0.16	0.02752	0.02223	0.01901	0.02106	0.02228	0.02263	0	100
0.2	0.03031	0.02427	0.02432	0.02287	0.02281	0.02244	0	100

Table 2. Mean errors with deterministic initialization. Mackey-Glass time series

	Hidden Neurones							
r_r	7	11	15	19	23	27	NP	%PP
0.04	0.02904	0.03086	0.03096	0.03109	0.03231	0.03295	45	91
0.08	0.01944	0.01860	0.01666	0.01565	0.01551	0.01585	0	100
0.12	0.02131	0.01742	0.01644	0.01607	0.01628	0.01602	0	100
0.16	0.02424	0.02029	0.01812	0.01729	0.01783	0.01809	0	100
0.2	0.02837	0.02083	0.01927	0.01874	0.02006	0.02111	0	100

In order to show that the deterministic initialization lead to an appropriate performance of RBNN when they are trained following the lazy learning approach, experiments with the lazy approach where the neurons centers are randomly initialized are also made. Table 1 shows the mean performance of the method for five random initializations, when RBNN with different number of hidden neurons are trained. Each value of the error for a specific number of neurons and radius corresponds to the mean value of five different mean errors. On the other hand, when the proposed deterministic initialization is applied, the obtained results are shown in table 2. We can observe that the error values are slightly better than the ones obtained when the neurons centers were randomly located. We must emphasize the advantage of this method where a single run is

needed whereas if the usual K-means algorithm is applied, several initializations must be made in order to ensure an adequate performance of the method.

The columns named "NP" displays the number of "null patterns", that is, test patterns for which the number of selected training patterns is zero. This situation might arise because of the dimensionality of the problem and the non-uniform distribution of data. The "PP" column displays the percentage of test patterns that are correctly answered ("Predicted Patterns"). As it is shown, when $r_r = 0.04$, there are 45 test patterns for which the networks can not make a prediction because the associated training sets are empty. Thus, these test patterns are discarded, corresponding the error values to the rest of patterns, that is, to the 91% of the whole test set.

We have applied the two alternative ways of treating these anomalous patterns are presented. *Method (a)*, that keeps the local approach,and *Method (b)* that renounce to the local approach and follows a global one. With the aim of studying the performance of both approaches, RBNN of different architectures are trained when a relative radius of 0.04 is taken. Both *Method (a)* and *Method (b)* have been applied and the obtained error values are shown in table 3, where we can see that method (b) behaves slightly worse than method (a) in all the cases. Thus, when a local approach is taken, the method gets better results than when all the available patterns are used to train the networks.

Table 3. Null patterns processing ($r_r = 0.04$). Mackey-Glass time series.

	Hidden Neurones							
	7	11	15	19	23	27	NP	%PP
Method (a)	0.02974	0.03043	0.03132	0.03114	0.03309	0.03373	45	100
Method (b)	0.03385	0.03641	0.03545	0.03464	0.03568	0.03408	45	100

As for the influence of the relative radius and the number of hidden neurons, it is possible to observe that the performance of the networks is scarcely influenced by the value of the relative radius when it is bigger than a certain value and the number of neurons is big enough. The mean error decreases with the radius until $r_r = 0.08$, and then it maintains its value nearly constant as the radius increases if the number of neurons is bigger than 7. Thus, the relative radius is not a critical parameter if the number of neurons is bigger than 7 and the relative radius is bigger than 0.08. When the number of neurons is small, the performance of the networks gets worse as the radius increases. This is explained because the number of training patterns selected is very big and the number of neurons of the RBNN are insufficient to fit such training set.

3.2 A Real Time Series Prediction Problem: The Venice Lagoon Time Series

The Venice lagoon time series represents the behavior of the water level at Venice lagoon. Unusually high tides result from a combination of chaotic climatic elements in conjunction with the more normal, periodic, tidal systems associated

with a particular area. The most famous example of flooding in the Venice lagoon occurred in November 1966 when, driven by strong winds, the Venice Lagoon rose by nearly 2 m. above the normal water level. That phenomenon is known as "high water" and many efforts have been made in Italy to develop systems for predicting sea levels in Venice and mainly for the prediction of the high water phenomenon [11].

There is a great amount of data representing the behavior of the Venice Lagoon time series. However, the part of data associated to the stable behavior of the water is very abundant as opposed to the part associated to high water phenomena. This situation leads to the following: the RBNN trained with a complete data set is not very accurate in predictions of high water phenomena. Hence, the aim in this context is to observe whether a selection of training patterns may help to obtain better predictions. A training data set of 3000 points corresponding to the water level measured each hour has been extracted from available data in such a way that both stable situations and high water situations appear represented in the set. High-water situations are considered when the level of water is not lower than 110 cm. 20 test patterns have also been extracted from the available data and they represent a situation when the water level is higher than 110 cm.

In order to apply LRBNN, different RBNN architectures have been trained during 500 learning cycles, and the relative radius has been fixed to different values from 0.04 to 0.2. As in the previous domain, two sets of experiments have been done: the first one corresponds to the usual, random K-means initialization; in order to obtain representative results, five runs of the method have been carried out and the mean of the results is showed in table 4.

The second set of experiments, carried out only once, corresponds to the deterministic initialization of the neurons centers. The results are displayed on table 5. As it happened on the previous domains, when the deterministic initialization of the centers is done, the results are similar or slightly better than when the centers are randomly located.

Table 4. Mean errors with random initialization of centers. Venice Lagoon time series

	Hidden Neurones							
r_r	7	11	15	19	23	27	NP	%PP
0.04	0.03596	0.03866	0.04035	0.04031	0.04015	0.04169	14	30
0.08	0.03286	0.03330	0.03150	0.03547	0.03799	0.03476	2	90
0.12	0.03219	0.02693	0.02490	0.02365	0.02677	0.02738	0	100
0.16	0.02487	0.02506	0.02554	0.02783	0.02600	0.02603	0	100
0.2	0.03350	0.03035	0.03094	0.03139	0.03155	0.03179	0	100

It is important to observe that there are null patterns even when the relative radius grows to 0.08. When $r_r = 0.04$, 14 test patterns, out of 20, can not be predicted. Thus, only a 30% of the test set can be properly predicted. And still for $r_r = 0.08$ 2 patterns are not predicted. The anomalous situations are now more frequent and this is explained as follows: the dimensionality of this problem

Table 5. Mean errors with deterministic initialization. Venice Lagoon time series.

	Hidden Neurones							
r_r	7	11	15	19	23	27	NP	%PP
0.04	0.03413	0.03378	0.03328	0.03465	0.03404	0.03389	14	30
0.08	0.03181	0.03028	0.03062	0.03041	0.03148	0.03017	2	90
0.12	0.02967	0.02682	0.02269	0.02234	0.02235	0.02643	0	100
0.16	0.02869	0.02398	0.02913	0.02059	0.02514	0.02552	0	100
0.2	0.03769	0.02420	0.02411	0.02728	0.02288	0.03336	0	100

is higher than the former one because this series problem has been modeled as a six-dimension function; besides, there are regions of the input space whose data density is very low. The test set has been generated so that its data represent the high water situations, and the training examples which corresponds to these unfrequent situations are very scarce. Thus, the null patterns processing methods presented in this work are essential in the LRBNN model. Table 6 shows the errors obtained when both methods (a) and (b) are applied if null patterns are found. It is important to realize that, although it seems that the results are worse than those seen on table 5, a 100% of the test patterns are properly predicted.

Table 6. Null patterns processing. Venice Lagoon time series.

		Hidden Neurones							
r_r	Meth	7	11	15	19	23	27	NP	%PP
0.04	(a)	0.06042	0.06276	0.06292	0.06186	0.06330	0.06352	14	100
0.04	(b)	0.09542	0.08128	0.06672	0.06239	0.06333	0.06500	14	100
0.08	(a)	0.03685	0.03447	0.03011	0.03197	0.02792	0.03231	2	100
0.08	(b)	0.04497	0.04382	0.03572	0.03407	0.03266	0.03441	2	100

In this domain, the differences between both methods are significant, specially when the relative radius is 0.04. In this case, 14 null patterns are found, that is, 70% of the whole test set. We can appreciate that method (a) achieves lower errors that method (b). Thus, when a lazy learning approach is applied the result is better than when the RBNN are trained with all the available training patterns.

It is possible to observe that, as in previous cases, when the relative radius is small, mean errors are high, due to the shortage of selected training patterns, and as the relative radius increases, the mean error decreases and then it does not change significatively. Thus, as it happened with the previous domains, the relative radius is not a critical parameter if the number of neurons and the relative radius are bigger enough.

3.3 Lazy Learning Versus Global Learning

In order to compare the lazy learning strategy (LRBNN) with the traditional one, RBNN with different number of hidden neurons (from 5 to 150) have been trained,

Table 7. Lazy learning versus traditional learning

	Mackey-Glass time series	Venice Lagoon time series
LRBNN	0.01551 ($r_r = 0.08$, 23 neurons)	0.02059 ($r_r = 0.16$, 19 neurons)
Traditional Method	0.10273 (110 neurons)	0.09605 (50 neurons)

in a global way, using the whole training data set in order to build a global approximation. In this work, the traditional learning has been carried out using a training and a validation data set, stopping the iterative process when both errors become stabilized. The standard K-means algorithm has been used and several experiments with different initial centers locations are made. In both approaches, the same data sets have been used. In table 7, the best results obtained in both domains for both methods, lazy and traditional ones, are shown. As it is possible to observe, in both domains the performance of the local method is significatively better than the performance of the traditional learning approach.

4 Conclusions

In this work, we try to complement the good characteristics of local and global approaches by using a lazy learning method for selecting the training set, using RBNN for making predictions. RBNN have some advantages: they are universal approximators and therefore the assumption of local linear behavior is no longer needed; besides, their training is very fast, without increasing significatively the computational cost of standard local learning approaches. We present a method (LRBNN) that can get the locality of the input space, and then uses a non-linear method to approximate each region of the input space. In addition, the selection of patterns is made using a kernel function, taking into account the distribution of data.

When a lazy learning strategy is used, two important aspects related to RBNN training and patterns selection have been taken into account. In the first place, the initialization of the neurons centers is a critical factor that influences RBNN performance. Usually, the initial location of centers are randomly established, but in a lazy strategy, in which a network must be trained for each new query, random initialization must be avoided. For this reason, in this work, the algorithm has been modified in a deterministic way to eliminate any initial condition influence with the objective of achieving the best performance. Regarding to the selection procedure, in which the Inverse kernel function is used, it may occur that no training pattern is selected for certain test patterns, due to the distribution of data in the input space. We have proposed and validated two different approaches to treat this problem.

LRBNN has been applied to two different domains: an artificial time series (the well known Mackey-Glass time series) and a real one (representing the Venice Lagoon water level). For both domains, we present the results obtained by LRBNN when a deterministic centers initialization is made. Besides, with the aim of showing the advantages of this deterministic initialization, the same method is applied but the RBNN are trained with a random initialization of their

centers. We show the mean results of several random initializations. As we said before, LRBNN provides two alternative ways of guarantying the selection of training examples for all the query instances. When the use of these alternative methods is necessary, the obtained results are also showed. Finally, LRBNN performance is compared with the performance of RBNN trained using a global approach, that is, using the whole training set.

The results obtained by LRBNN improves significatively the ones obtained by RBNN trained in a global way. Besides, the proposed deterministic initialization of the neurons centers produces similar or slightly better results than the usual random initialization, being thus preferable because only one run is necessary. Moreover, the method is able to predict 100% of the test patterns, even in those extreme cases when no train examples would be selected using the normal selection method. The experiments show that the relative radius, parameter of the method, is not a critical factor because if it reaches a minimum value and the network has a sufficient number of neurons, the error on the test set keeps its low value relatively constant.

Thus, we can conclude that the combination of lazy learning and RBNN, can produce significant improvements in some domains.

Acknowledgments. This article has been financed by the Spanish founded research MEC project OPLINK::UC3M, Ref: TIN2005-08818-C04-02

References

1. D.W. Aha, D. Kibler, and M.K. Albert. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 1991.
2. L. Bottou and V. Vapnik. Local learning algorithms. *Neural Computation*, 4(6):888–900, 1992.
3. C.G. Atkenson, A.W. Moore, and S. Schaal. Locally weighted learning. *Artificial Intelligence Review*, 11:11–73, 1997.
4. B.V. Dasarathy (Editor). Nearest neighbour(NN) norms: NN pattern classification techniques. *IEEE Computer Society Press*, 1991.
5. J. Ghosh and A. Nag. *An Overview of Radial Basis Function Networks.* R.J. Howlett and L.C. Jain (Eds). Physica Verlag, 2000.
6. J.E. Moody and C. Darken. Fast learning in networks of locally tuned processing units. *Neural Computation*, 1:281–294, 1989.
7. J. Park and I.W. Sandberg. Universal approximation and radial-basis-function networks. *Neural Computation*, 5:305–316, 1993.
8. J.M. Valls, I.M. Galván, and P. Isasi. Lazy learning in radial basis neural networks: a way of achieving more accurate models. *Neural Processing Letters*, 20:105–124, 2004.
9. D. Wettschereck and T. Dietterich. Improving the perfomance of radial basis function networks by learning center locations. *Advances in Neural Information Processing Systems*, 4:1133–1140, 1992.
10. L. Yingwei, N. Sundararajan, and P. Saratchandran. A sequential learning scheme for function approximation using minimal radial basis function neural networks. *Neural Computation*, 9:461–478, 1997.
11. J.M. Zaldívar, E. Gutiérrez, I.M. Galván, F. Strozzi, and A. Tomasin. Forecasting high waters at Venice Lagoon using chaotic time series analysis and nonlinear neural networks. *Journal of Hydroinformatics*, 2:61–84, 2000.